# How to Use Visual Basic to Interface Scientific Instruments to a Personal Computer

**Nikos Papadopoulos* and Maria Limniou**

*Laboratory of Physical Chemistry, Department of Chemistry, Aristotle University, 54006 Thessaloniki, Greece, npapado@chem.auth.gr*

*Received April 19, 2002. Accepted May 9, 2002.*

**Abstract:** Computer-controlled instruments are expensive and for this reason are not widely used in student laboratories. In this paper we describe how to interface a data-acquisition board using Microsoft Visual Basic. Visual Basic can be used to quickly develop an application in the Microsoft Windows environment. A single computer system can serve as several different measuring systems by simple adaptation of the software.

## Introduction

Scientific instruments controlled by built-in microprocessors or through connections to accompanying microcomputers can be found in government, research, and industrial laboratories. Students should have an early acquaintance with computer-controlled instrumentation so that they learn what can be conveniently done using a computer.

Computer-controlled instruments are expensive and for this reason are not widely used in student laboratories. A laboratory interfacing system using a personal computer provides a relatively inexpensive alternative. Up to now the principal obstacle in building homemade microprocessor-controlled instrumentation has been the time and effort needed to create the software [1–3].

To perform any task, microprocessors must be programmed using mathematical and logical operations. Program development using a character-based programming language is tedious and time-consuming. The most difficult part of writing an application program in a conventional language is the creation of a user-friendly interface, that is, a user interface that is readily identifiable and easy to understand. Today, there are graphics-program environments that make it easy to build the front panel of a "virtual instrument" with its knobs, switches, graphs and displays on a computer screen [4–13]. Rather than having to write numerous lines of code to describe the appearance and location of interface elements, we can simply add prebuilt objects into place on the screen. Visual Basic is an environment that provides the necessary tools to build very rapidly user-friendly interfaces in the Windows operating system. The applications that are created run on Windows PCs without additional licensing.

Some years ago, interfacing an analog instrument to a digital computer required considerable expertise in both analog and digital electronics. Now, the situation has changed. Today, it is possible to purchase very satisfactory, off-the-shelf interface cards for a personal computer at reasonable cost. The interface board that we have used in this work is the super 14-bit A/D-D/A card from Decision Computer International, Taipei, which is an ISA slot interface card and is suitable for rather old computers. More information on this card can be found at the companies Internet site [14].

## How to Control the Digital to Analog Converter

The super 14-bit A/D-D-A data acquisition card plugs into one of the PC's expansion slots, provides an accuracy of 14 bit and contains 16 single-ended channels for analog-to-digital conversion and 2 channels for digital-to-analog conversion. This cards offers two jumper-selectable options for the voltage output of each channel: unipolar or bipolar output swing and 2.5, 5, or 10 V reference voltage that is derived from an on-board reference, which is either bipolar or unipolar .The voltage input range is also jumper-configurable for unipolar or bipolar operation and for 2.5, 5, or 10 V input range .The manufacturer gives a conversion time less than 2 μs for digital-to-analog operation and a conversion time less than 28 μs for an analog-to-digital operation. The super 14-bit A/D-D-A card has several jumper switches that must be set to define its operation. These are used to select the card's D/A output range, its A/D input range, and its I/O base address. The card uses 16 consecutive address locations in the PC's I/O space. Interface cards with similar characteristics can be obtained commercially from many different manufacturers [15].

The following example illustrates how easy it can be to create a useful instrumentation application using Visual Basic. To demonstrate how this is done, we describe the steps used to create a simple application that controls the output of the digital-to-analog converter of the interface card. The application consists of a label and a scroll bar. When the user moves the scroll bar, the numerical value on the label changes from 2.500 to –2.500 V and at the same time this voltage value appears at the output of the digital-to-analog converter. The label and the scroll bar are placed in a form. For example in the illustration in Figure 1, the voltage output of the D/A converter is at a setting of +0.882 V on a scale of +2.500 to –2.500 V.

The form acts as a window that permits the program operator to communicate with the computer and the instrument to which it is interfaced. Rather than writing numerous lines of code to describe the appearance and location of interface elements, we simply add prebuilt objects into place on screen. Visual Basic provides a set of tools that we use at design time to place controls on a form. Forms and controls are the basic building blocks used to create the user interface, the visual part of the application with which the user will interact. We use the

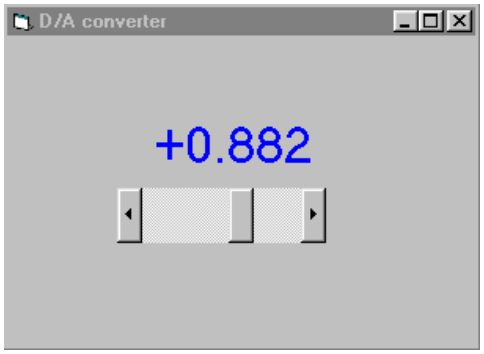


**Figure 1.** A simple application

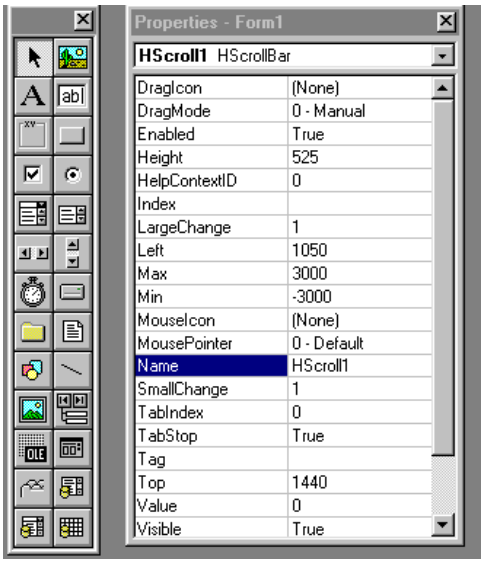


**Figure 2.** The toolbox and the properties window.

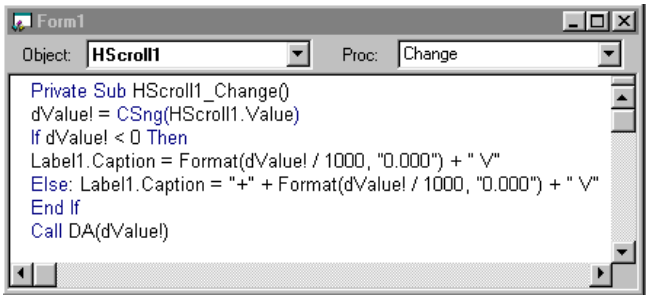


**Figure 3.** The Hscroll_change event subroutine.

toolbox to draw a scroll bar and a label on the form (See Figure 2).

Each form and control in Visual Basic has a predefined set of events. If one of these events occurs and there is code in the associated event procedure, Visual Basic invokes that code. For example, whenever the user moves the scroll bar, at run time the change event procedure (Figure 3) is invoked. In traditional or procedural applications, the application itself controls which portions of code will execute and in what sequence [16–19]. Execution starts with the first line of code and follows a predefined path throughout the application, calling procedures as needed. In an event-driven application, the code does not follow a predetermined path, instead it executes different code sections in response to events. Events can be triggered by the user's actions, by messages from the system or other applications, or even from the application itself. The sequence of these events determines the sequence in which the code executes; thus, the path through the application's code differs each time the program runs. Each control on a form has a corresponding set of event procedures associated with it.

We place code in the change event procedure (Figure 3) to control the output of the digital-to-analog converter of the interface card. The text displayed in the label is controlled by the label's caption property, which can be set at design time in the properties window or at run time by assigning it in code. We can set properties to specify the appearance, operation, labeling, range, precision, and format of the controls. Once we have configured the properties of a control we can programmatically access or change the values of any property from within our program.

In the DOS version of the Basic programming language the inp function and the out command gave the user the ability to send and return bytes to and from the hardware I/O ports. Visual Basic's repertoire of functions and commands does not include equivalent functions. Visual Basic has no built-in way to access ports; however, Visual Basic accepts access functions and procedures in the form of dynamic link libraries (DLL) that are written in other programming languages or are included in the windows operating system. The program can communicate with the card through a DLL that emulates the inp function and the out commands of Basic in the Visual Basic programming environment. DLLs with this functionality can be found in the Internet for 16-bit and 32-bit environments. The DLL file that gives input–output capability to Visual Basic must be included in the system subdirectory of the Windows file. These DLLs, and much more, can be found at the Internet site in reference 20.

Many manufacturers of plug-in digital-to-analog acquisition boards provide Visual Basic custom controls for data acquisition. These custom controls provide the drivers for the interface card. An interface-card driver contains high-level functions for controlling the specific acquisition board so that it is not necessary to write a program to access the cards' registers. Many software malfunctions stem from poor driver performance or poor driver documentation. Sometimes it is preferable to write and test your own driver for the interface card and find out what are the limitations imposed by the hardware. This is not a hard task. The user manuals of most interface cards provide detailed information about the steps required to perform an analog-to-digital conversion. With the exception of very fast phenomena, all of these steps can be coded in Visual Basic without sacrificing measurement accuracy. We give as an example (Figure 4) the subroutine that can be used to drive the super 14-bit A/D-D/A card.

In order to trigger a digital-to-analog conversion with the super 14-bit A/D-D/A card we must output a value between 0 (–2.5 V) and 16383 (2.5 V). This value must be separated into a high byte and a low byte, and then the high byte must be output port address &H165 (in hexadecimal) and the low byte to the address &H164 (in hexadecimal). We have written a subroutine that takes a numerical value as input and instructs the digital-to-analog converter to translate this value to an output voltage. The digital-to-analog converter can be used to generate simple waveforms such as sine square and triangle waves or more complex completely user-defined waveforms.

DLL procedures reside in files that are external to our application, and for this reason we must specify where the

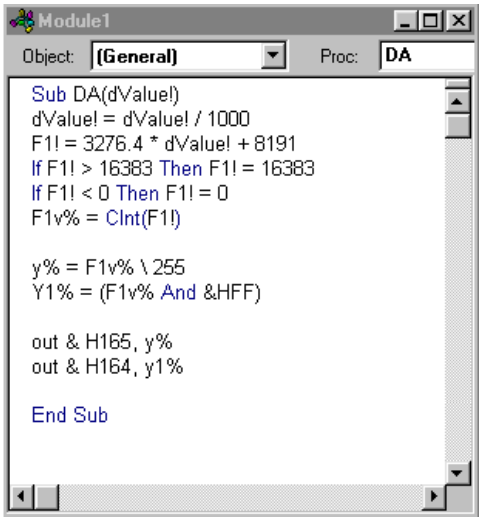


**Figure 4.** The digital to analog conversion subroutine.

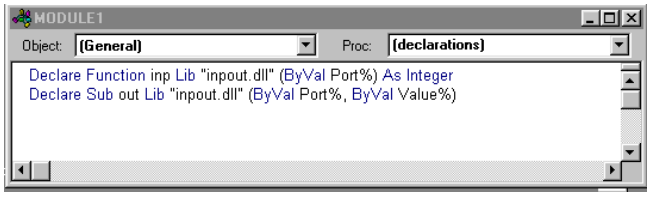


**Figure 5.** The declare statements for the input DLL.

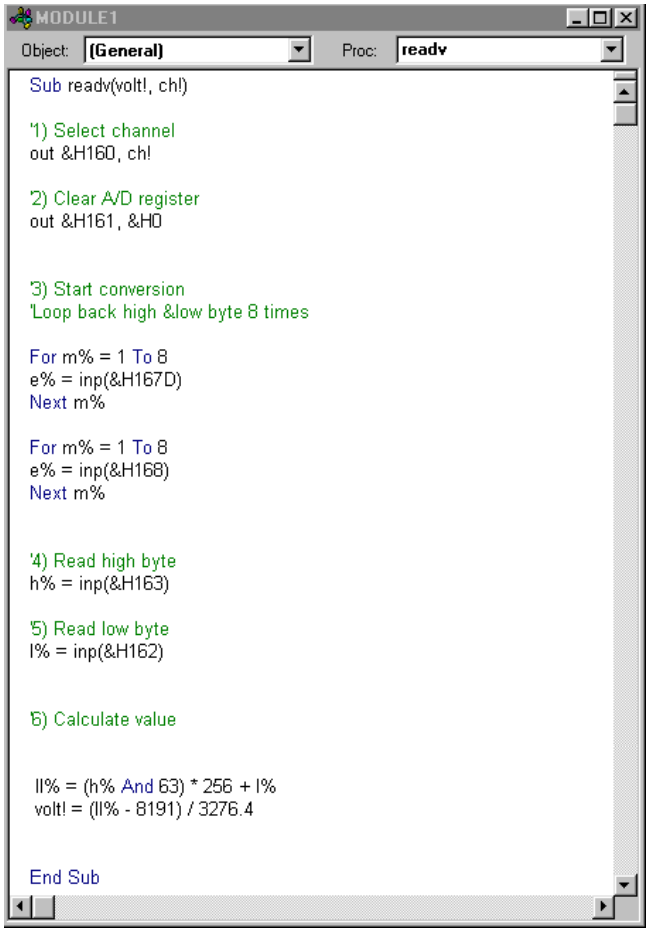


**Figure 6.** The analog-to-digital-conversion subroutine.

procedures are located and identify the arguments with which they should be called. We provide this information with a declare statement (Figure 5). Once we have declared a DLL procedure, we can use it in our code just like a native Visual Basic procedure.

One of the usual tasks that students perform in an experiment is to gather data for the study of a chemical phenomenon. Usually the students collect the data manually, and then they plot the data to create a visual representation of the numerical data. A data-acquisition board can be used to acquire data directly from laboratory equipment. As old-model instruments are gradually replaced by newer models, the old instruments are usually abandoned. These instruments can be used in the student laboratory. Older analog instruments usually have a connection for a strip-chart recorder. The instrument's output must often be conditioned to provide signals suitable for the digital-to-analog-conversion board. External circuitry can improve the accuracy of the data. Amplifiers can boost the level of the input signal to better match the range of the digital-to-analog converter, thus increasing the resolution and sensitivity of the measurement. Additionally, external circuitry can include analog filters to reject unwanted noise. More information on hardware on signal conditioning can be found at National Semiconductor's Web site [21].

Code in Visual Basic is stored in modules. Simple applications can consist of just a single form, and all of the code in the application resides in that form module. Code that is not related to a specific form or control can be placed in a different type of module, a standard module. Standard modules are containers for procedures and declarations commonly accessed by other modules within the application. They can contain global (available to the whole application) or module-level declarations of variables, constants, types, external procedures, and global procedures. The code that we write in a standard module is not necessarily tied to a particular application; if we are careful not to reference forms or controls by name, a standard module can be reused in many different applications.

### Analog to Digital Conversion

The following example describes the steps used to build a simple data-acquisition application with the super 14-bit A/D-D-A card. In order to trigger an analog-to-digital conversion with this card we must do the following:

(1) select the channel,
(2) clear the A/D register,
(3) start the conversion,
(Loop back the high and low byte eight times.)
(4) read the high byte,
(5) read the low byte,
(6) calculate the value.

A computer-controlled acquisition system (such as in Figure 7) samples data at a user-specified rate. Visual Basic provides a timer control that uses the internal timer of the computer that ticks about 18.2 times a second. We use this timer as the time base of our recorder. Data is sampled continuously at a rate of about a data point every 50 ms. With this sampling rate about 20 points are sampled every second. This relatively slow sampling rate is adequate for most student experiments, and sometimes it is even desirable to work with a lower sampling

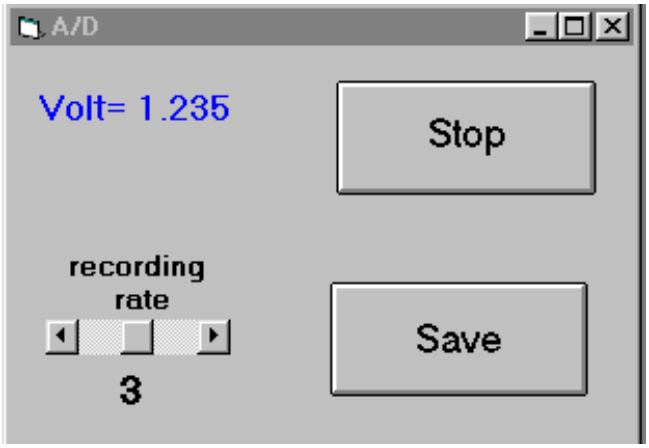


Figure 7. A data-acquisition application

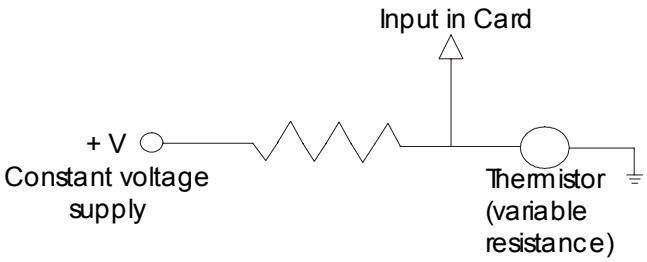


**Figure 8.** A typical microcomputer's sensory interface.

rate. Information on how to obtain faster sampling rates can be found at the Web sites in reference 22.

At the end of the experiment the student can save the experimental data on a disk in a sequential file. Data collected in the form of sequential or random-access files from Visual Basic are directly compatible with many spreadsheet programs, for example, Excel. Computer programs can be written to facilitate complicated calculations required to derive meaningful values from raw experimental data. Data treatment can be accomplished with a commercial software package and the students can take advantage of the very good graphics and hard copy in the forms. It also gives an opportunity for students, while they are treating their results, to gain some experience with data-handling techniques and the use of the spreadsheet programs.

Sensory interfaces (Figure 8) are often used in the student laboratories. Sensors provide information in the form of an electrical signal related to the physical or chemical environment. A thermistor can be interfaced to a digital computer to follow a variety of temperature-versus-time experiments in general chemistry [23].

## References and Notes

1. Muyskens, M. *J. Chem. Educ.* **1997,** *74* ( 7), 850.
2. Roldan, E.; Dominquez, M.; Arjona, D. *Computers & Chemistry* **1986,** *10* (3), 187.
3. He, P.; Faulkner, L. *J. Chem. Inf. Comput. Sci.* **1985,** *25,* 275–282.
4. Baumann, M. *Computers & Education* **2001,** 36 (3), 245–264.
5. David, F.; Papadopopoulos, N. *Electoanalysis* **1991,** *3,* 721.
6. David, F.; Ougenoune, H.; Bolios, A.; Papadopoulos, N. *Anal. Chim. Acta* **1994,** *292,* 297–304.
7. Spanoghe, P;.Cocquyt, J; Van der Meeren, P.; *J. Chem. Educ.* **2001,** *78* (3), 338.
8. Gostowski, R. *J. Chem. Educ.* **1996,** *73* (12), 1103.
9. Drew, M. S. *J. Chem. Educ.* **1996,** *73* (12), 1107.
10. Muyskens, A. M.; Glass, S. V.; Wietsma, T. W.; Gray, T. M.; *J. Chem. Educ.* **1996,** *73* (12), 1112.
11. Ogren, P. J.; Jones, T. P.; *J. Chem. Educ.* **1996,** *73* (12), 1115.
12. Allerhand, A.; Galuska, A. *Chem. Educator* [Online] **2000,** 5 (2), 71–76; DOI 10.1007/s00897990368a.
13. Ritter, D; Johnson, M. *J. Chem. Educ.* **1997,** *74,* 120.
14. Decision Computer. www.decision.com.tw (accessed Aug 2002).
15. (a) Datal. http://store.datel.com (accessed Aug 2002); (b) Advantech. www.advantech.com (accessed Aug 2002); (c) Delta Tech. www.delta-technical.com (accessed Aug 2002).
16. Halvorson, M. *Step by Step (Microsoft Visual Basic 6.0)*; Microsoft Press: Redmond, WA, 1998.
17. Eidahl, L. D. *Edition Using Visual Basic 6;* QUE Corporation: Indianapolis, IN, 1999.
18. Petroutsos, E. *Visual Basic 6;* SUBEX: Berkeley, CA 1998.
19. Kofler, M. *Visual Basic Database Programming;* Addision-Wesley: Reading, MA, 2002.
20. Axelson, J. Parallel Port Central. http://www.lvr.com/parport.htm (accessed Aug 2002).
21. National Semiconductor, Application Notes and Other Documents. http://www.national.com/apnotes/ (accessed Aug 2002).
22. (a) Stopuhr. http://www.activevb-archiv.de/vb/VBtips/VBtip0011.shtml (accessed Aug 2002); (b) vbapi.com—part of the VB-World Network http://www.vbapi.com/ref/funcc.html#timers (accessed Aug 2002).
23. Wong, S.; Popovich, N. D.; Coldiron, S. J. *J. Chem. Educ.* **2001,** *78* (6), 798.